(2)

MASSACHUSETTS INSTITUTE OF TECHNOLOGY VLSI PUBLICATIONS

VLSI Memo No. 89-543
May 1989

Dynamic Embedding of Trees in Hypercubes with Constant Dilation and Load

Tom Leighton, Mark Newman, and Eric Schwabe

Abstract

Parallel computations often yield computation structures which are trees; the shape of such a tree evolves over time as the computation progresses. However, parallel computers are usually designed as networks of processors with fixed connections; it is therefore important to embed the dynamic structure of a computation efficiently in a fixed network. We consider the problem of dynamically embedding an evolving binary tree with at most N nodes in an N-node hypercube. We present a simple randomized algorithm which uses only local control and guarantees constant dilation, while maintaining constant load with high probability; this is the first load-balancing algorithm which achieves constant dilation. We also prove that random solutions to this problem are highly desirable, by demonstrating that any deterministic embedding algorithm which maintains constant load must have $\Omega(\sqrt{\log N})$ dilation.

Microsystems Research Center
Room 39-321

Massachusetts Institute of Technology

Cambridge Massachusetts 02139

Telephone (617) 253-8138

Acknowledgements

To appear in the *ACM Symposium on Parallel Algorithms and Architectures*, June 1989. This research was supported in part by the Defense Advanced Research Projects Agency under contract numbers N00014-80-C-0622 and N00014-87-K-0825, the Air Force under contract number AFOSR-86-0078, and the Army under contract number DAAL-03-86-K-0171. Leighton was supported in part by a NSF Presidential Young Investigator Award with matching funds from IBM. Schwabe was supported in part by a NSF graduate fellowship.

Author Information

Leighton: Department of Mathematics and Laboratory for Computer Science, Room NE43-332, MIT, Cambridge, MA 02139. (617) 253-5876.
Newman: Department of Mathematics and Laboratory for Computer Science, Room NE43-334, MIT, Cambridge, MA 02139. (617) 253-5866
Schwabe: Department of Mathematics and Laboratory for Computer Science, Room NE43-328, MIT, Cambridge, MA 02139. (617) 253-1365.

Dynamic Embedding of Trees in Hypercubes with Constant Dilation and Load

(extended abstract)

Tom Leighton *
Mark Newman †
Eric Schwabe ‡

MIT Department of Mathematics
and
Laboratory for Computer Science
Cambridge MA 02139

Abstract

Parallel computations often yield computation structures which are trees; the shape of such a tree evolves over time as the computation progresses. However, parallel computers are usually designed as networks of processors with fixed connections; it is therefore important to embed the dynamic structure of a computation efficiently in a fixed network. We consider the problem of dynamically embedding an evolving binary tree with at most N nodes in an N-node hypercube. We present a simple randomized algorithm which uses only local control and guarantees constant dilation, while maintaining constant load with high probability; this is the first load-balancing algorithm which achieves constant dilation. We also prove that random solutions to this problem are highly desirable, by demonstrating that any deterministic embedding algorithm which maintains constant load must have $\Omega(\sqrt{\log N})$ dilation.

1 Introduction

Solving large problems efficiently in parallel sometimes leads to a computation which has a tree structure. Examples of this are divide-and-conquer and branch-and-bound computations

*Supported by Air Force contract AFSOR-86-0078, DARPA contract N00014-80-C-0622, Army contract DAAL03-86-K-0171, and an NSF Presidential Young Investigator Award with matching funds from IBM.
†Supported by DARPA contract N00014-80-C-0622.
‡Supported by an NSF Graduate Fellowship

[KZ]. In general, the resulting computation tree is *dynamic* in that the tree grows and shrinks at its leaves in an unpredictable way.

However, large parallel computers are built with a fixed set of connections: the neighbors of a processor do not change over time. Therefore we must map the nodes of a dynamic tree to the processors of a fixed-connection network. In order to insure fast communication between tree nodes, adjacent nodes in the tree should be mapped to processors which are near each other; in order to insure that full advantage is taken of the available processing power, the nodes of the tree should be spread out evenly in the network so that no single processor has to simulate disproportionately many tree nodes. This leads us to the question of how to *efficiently* embed dynamic trees in a fixed-connection network.

In this paper we consider the problem of embedding dynamically growing rooted binary trees in a hypercube network. There are several measures of the efficiency of an embedding. One is *dilation*, defined to be the maximum distance in the hypercube between two nodes which are adjacent in the embedded tree. Low dilation means that each tree node is close to its parent, facilitating communication between them. Another important measure is the *load* of an embedding, which is the maximum number of tree nodes mapped to a single node of the hypercube. A good embedding will have a load which is low, implying an even distribution of tree nodes and therefore efficient use of the available processors. Note that this problem has been solved any *fixed* binary tree, as it is known that any N-node binary tree can be embedded one-to-one in an N-node hypercube with constant dilation [BCLR]; our results show that these properties can be maintained as the tree evolves.

In this paper we present a simple randomized algorithm to embed a dynamically evolving binary tree with at most N nodes in an N-node hypercube. The algorithm is distributed and requires only local control; constant dilation is guaranteed, and with high probability, the resulting embedding will also maintain constant load. This work is closely related to recent work of Bhatt and Cai [BC]; in their paper they present a randomized algorithm for this problem which guarantees $O(\log \log N)$ dilation, and achieves constant load for $O(N)$-node trees with high probability. Though at first glance the algorithm in this paper may seem nearly identical to that of Bhatt and Cai, subtle differences in the use of randomness lead to a provably stronger result. In addition, we prove that no deterministic embedding algorithm which maintains constant load can achieve dilation better than $\Omega(\sqrt{\log N})$, even for trees which only grow. This emphasizes the desirability of randomized solutions to this problem.

The rest of the paper is organized as follows. Section 2 contains preliminaries and definitions along with a description of the embedding algorithm. Section 3 contains the analysis of our algorithm. Section 4 presents the negative result for deterministic algorithms.

2 Preliminaries

Let $\mathcal{T}$ be a rooted binary tree which is growing and shrinking over time. By this we mean that in each time step some nodes of $\mathcal{T}$ decide to grow children and some leaves cease to

exist. Initially, $\mathcal{T}$ consists of only the root. We assume that $\mathcal{T}$ never has more than N nodes and that it grows and shrinks over a period of time whose length is polynomial in N.

Let H_n denote the n-dimensional hypercube with $N = 2^n$ nodes, represented by the set of all strings in $\{0,1\}^n$. The $n2^{n-1}$ edges of H_n connect nodes with strings differing in precisely one bit. The *dimension* of an edge is the bit position (0 to $n-1$) in which the incident nodes differ. If $x \in H_n$, then x' refers to the node adjacent to x and differing from it in the i^{th} dimension bit.

An *embedding* of $\mathcal{T}$ into H_n is a map $\phi : \mathcal{T} \mapsto H_n$ that maps nodes of $\mathcal{T}$ to nodes of H_n. For two hypercube nodes x and y, let $d(x,y)$ be the Hamming distance from x to y; i.e. the number of bit positions where x and y differ. The *dilation* of an embedding is $max\{d(\phi(v),\phi(w))\}$, where the maximum ranges over all v and w which are neighbors in the tree. The *load* of an embedding equals $max\{\phi^{-1}(x)|x \in H_n\}$, the maximum number of nodes in $\mathcal{T}$ mapped to a single node of H_n.

For any node $x \in H_n$, the *star centered at* x is the set of nodes consisting of x itself and all neighbors of x. Each star contains $n+1$ nodes. We present an algorithm which dynamically embeds the tree $\mathcal{T}$ in H_n such that all stars have $O(n)$ tree nodes mapped to them. We will then show how to slightly alter this embedding (dynamically) so that the resulting load is constant. Throughout, the dilation will be constant.

Fix an arbitrary constant b. (For ease of exposition, we assume that b divides n.) Call the distance from a tree node v to the root the *depth* of v. The embedding algorithm maps tree neighbors to hypercube nodes which are at distance at most b.

```
for each node v which grows a child w
    y ← φ(v)
    let v be at depth i
    for j ← ib (mod n) to (i + 1)b − 1(mod n)
        y ← y^j with probability 1/2
    φ(w) ← y
```

Figure 1: The Embedding Algorithm

3 Analysis of the Algorithm

We first discuss the case in which $\mathcal{T}$ is only growing. We will show later that we can handle a tree which shrinks as well.

Theorem. *If a growing tree $\mathcal{T}$ with at most N nodes is dynamically embedded in the hypercube H_n according to the rule given above with $b \geq 8$, then with high probability no more than $O(n)$ tree nodes are mapped into any star of H_n.*

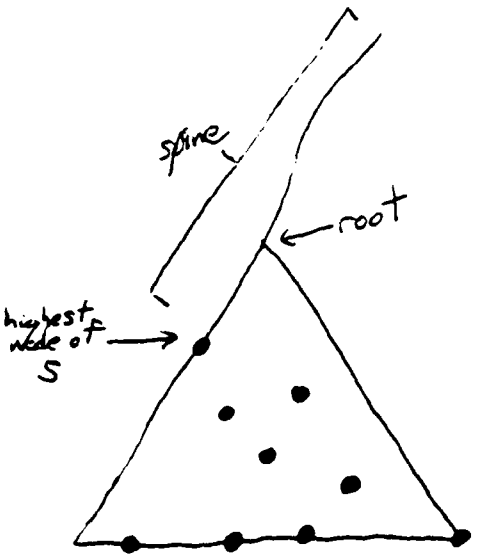

Figure 2: An Extended Tree

Fix a node $x_0 \in H_n$. Call a node $v \in \mathcal{T}$ a *star node* if $\phi(v)$ lies in the star centered at x_0. We will prove that with high probability there are only $O(n)$ star nodes. If we let x_0 range through the N nodes in H_n, this proves the theorem as well.

Two tree nodes v and w are *close* if $d(u,v), d(u,w) \leq \frac{n}{5}$, where u is the lowest common ancestor of v and w. A set S of tree nodes is a *close set* if for any two nodes $v, w \in S$ there is an ordered set $< u_i >$ of elements of S where $v, w \in \{u_i\}$ and u_i is close to u_{i+1} for all i. The *highest node* of a close set is that node in the set at the lowest depth in the tree. If there are more than one at the lowest depth, the highest node is a fixed arbitrary one of these.

We form the *extended tree* T of a close set S as follows. First we include all the edges which lie on the (unique) shortest path between any two close nodes in S. By the definition of a close set, the edges between all pairs of close nodes form a tree. We refer to the (unique) highest node of this tree as the root of the extended tree. Say that the path from the highest node of S to the root of the extended tree has length l. Then we include the first $\frac{n}{5} - l$ edges on the (unique) shortest path from the root of the extended tree to the root of $\mathcal{T}$. If we reach the root of $\mathcal{T}$ before traversing $\frac{n}{5} - l$ edges, we stop. These new edges, the tree already defined, and all incident vertices form the extended tree of S. The extended tree of S is a subtree of $\mathcal{T}$ and its leaves are all elements of S. The path in the extended tree which contains ancestors of the highest node of S (but not the highest node itself) is the *spine* of the extended tree. The highest node in the spine is the *tip* of the spine.

Let $< S_i >$ be the indexed family of maximal close sets of star nodes. By calling these sets maximal we mean that the union of any two of them is not a close set. Let $< T_i >$ be the associated family of extended trees.

Lemma 3.1. *No two trees T_i and T_j share a node.*

Proof. Any node u in an extended tree has a close star node descendent. Thus if u were

in two trees, it would have a close star node descendent in each tree and these nodes would therefore be close to each other. In that case, $S_i \cup S_j$ would be a close set, violating our assumption. ∎

Lemma 3.2. *At most one extended tree does not include the last $\frac{n}{b}$ edges along the shortest path from the root of $\mathcal{T}$ to the highest node in the associated close set.*

Proof. Any extended tree which did not include those edges would include the root of $\mathcal{T}$. By lemma 3.1, there can be only one such tree. ∎

Say that extended tree T_i has highest node v_i, contains m_i total star nodes and that the path from v_i to the root r_i of T_i has length l_i. Because the root is the lowest common ancestor of all the star nodes in the close set, either both of its children have star node descendents or r_i is a star node itself. If r_i is not a star node then there is another star node v_i' whose path to r_i does not intersect the path from v_i to r_i. Furthermore, this path has at least $l_i - 1$ nonstar internal nodes since otherwise v_i' would be closer to r_i than is v_i. If r_i is a star node then r_i is also the highest node and $l_i = 0$. In this case $l_i - 1 = -1$ easily bounds the number of nonstar nodes off the spine. Thus in the extended tree there must be at least the nodes on the spine, the m_i star nodes, and the $l_i - 1$ nonstar nodes off the spine accounted for above. All other nodes we denote as surplus; let s_i be their count. In the extended tree T_i we have a total of $m_i + l_i + s_i - 1$ nodes not on the spine.

Lemma 3.3. *Over all ways to choose which nodes in $\mathcal{T}$ are star nodes, no more than $2^{3m_i+4l_i+3s_i-7}$ different extended trees may be formed with highest node v_i, m_i total star nodes, $d(v_i, r_i) = l_i$ and $m_i + l_i + s_i - 1$ nodes off the spine.*

Proof. The nodes on the spine are fixed by v_i. All other nodes in the tree descend from the nodes on the spine. We can grow any extended tree from this spine. Maintain a queue of extended tree nodes. Initialize the queue with the nodes on the spine. At any step, if the node at the head of the queue has children in $\mathcal{T}$, decide which of them are to be included in the extended tree. Add to the queue those children included in the extended tree. Once we have processed $m_i + l_i + s_i - 2$ nodes we may stop. The one remaining node in the queue has no children in the extended tree.

Each of the first $l_i - 1$ nodes on the spine has at most one child not on the spine. For each node we can choose whether or not her child belongs to the extended tree. There are at most 2^{l_i-1} ways to do this. The l_i^{th} node along the spine is the root of the extended tree. Her other child must be in the tree. All other nodes on the spine also have no choice: their other children surely do not belong to the tree. Any node not on the spine has at most two children and thus at most four choices for including her children in the tree. Thus the total number of ways to grow the remainder of the tree is no more than $4^{m_i+l_i+s_i-2}$.

Finally, we choose which of the nodes in the extended tree are star nodes. These cannot lie on the spine. The node v_i certainly is a star node. We must choose $m_i - 1$ of the other nodes to be star nodes. There are no more than $2^{m_i+l_i+s_i-2}$ ways to do this.

Many of these choices do not result in valid extended trees. First, v_i might not be the highest node of the extended tree formed. Second, we might not choose all the leaves as star

nodes. However, we have overcounted at worst. Multiplying the three factors bounds the number of ways to specify the shape of the extended tree and to choose the star nodes in it. ∎

Lemma 3.4. *Let T_i be an extended tree with highest node v_i, m_i total star nodes, $d(v_i, r_i) = l_i$ and $m_i + l_i + s_i - 1$ nodes off the spine. The probability that T_i results during the embedding of T is no more than $\frac{n+1}{N}(2^{-b}(b+1))^{m_i+l_i+s_i-2}$ if it does not contain the root and at most $(2^{-b}(b+1))^{m_i+l_i+s_i-2}$ if it does.*

Proof. No matter where the tip of the spine is embedded, the coordinates changed along the path to v_i must force $\phi(v_i)$ to lie in the star. If the path is of length $\frac{n}{b}$, then each coordinate has exactly one chance to change. Thus v_i is equally likely to be mapped anywhere in H_n. The probability v_i is mapped into the star is precisely $\frac{n+1}{N}$. Otherwise, we bound the probability that v_i is a star node by 1. Deciding how these coordinates change determines the embedding of all nodes in the spine.

Consider a path of length $p \leq \frac{n}{b}$ from a node u to the star node v along which no other nodes are star nodes. Say that the embedding $\phi(u)$ has already been determined, but that the coordinate changes in all edges along the path have yet to be decided. There are pb coordinates considered along the path. If $\phi(u)$ agrees with x_0 in all the other coordinates, we might choose exactly one of the pb path coordinates to differ from that of x_0 or we might choose them all to be the same. If we choose more than one to differ, then v cannot map into the star, a contradiction. If $\phi(u)$ and x_0 do not agree in the nonpath coordinates, then there is at most one setting of the pb coordinates which would lead $\phi(v)$ to lie in the star. In either event, the probability that the coordinates were chosen in a way consistent with the tree structure (i.e. which nodes are star nodes) is at most $\frac{pb+1}{2^{pb}} \leq (2^{-b}(b+1))^p$.

Recall that we have determined the embeddings of the nodes on the spine. Arbitrarily dissect the remaining edges into paths of length at most $\frac{n}{b}$ which end at star nodes but have no star nodes internal to them. We can do this because every node which is not a leaf of the extended tree has a star node at most distance $\frac{n}{b}$ below it. (A leaf which is not a star node is not on a close path between star nodes and thus would not have been included in the extended tree.) We can then analyze the coordinate changes which occur in the subtree below any node in the spine. If we start with the node on the spine, then we can consider the paths in turn so that the embedding of the first node in the path has already been determined but nothing else along the path has been determined. Coordinate changes along any path are independent of what has been determined before. One bound on the probability that the coordinate changes in the paths actually result in the mapping of all star nodes into the star is the product of the bounds on the probabilities for the individual paths. The sum of the lengths of all the paths equals the number of edges not in the spine. Thus we may use the upper bound found in the previous paragraph to bound the probability that all coordinate changes produce the desired structure of the tree. This probability is no more than $(2^{-b}(b+1))^{m_i+l_i+s_i-2}$. ∎

We may now bound the probability that certain structures involving star nodes arise during the embedding.

Lemma 3.5. *Fix t, the number of extended trees formed by the close sets of star nodes and index the trees arbitrarily as $< T_i >$. Choose a node v_i to be the highest node of T_i and m_i to be the number of star nodes in T_i. Let $m = \sum_i m_i$. Further, choose l_i and s_i for T_i. Then if we use the algorithm shown above to embed $\mathcal{T}$ in H_n, the probability that the resulting extended trees have the parameters chosen is at most*

$$\left(\frac{n+1}{N}\right)^{t-1}(2^{2b-7}(b+1)^{-2})^t(2^{3-b}(b+1))^m\prod_i(2^{4-b}(b+1))^{l_i}(2^{3-b}(b+1))^{s_i}$$

Proof. Fix a particular family of extended trees satisfying the chosen parameters. As $\mathcal{T}$ is embedded, we know by lemma 3.1 that the choices made on the edges of any tree are independent of the choices for other trees. Thus to bound the probability that the family of extended trees arises from the embedding, we need only multiply the probability bounds for the individual trees. This product is at most $(\frac{n+1}{N})^{t-1}\prod_i(2^{-b}(b+1))^{m_i+l_i+s_i-2}$ by lemmas 3.2 and 3.4.

The number of different families of trees $< T_i >$ satisfying the chosen parameters is at most $\prod_i 2^{3m_i+4l_i+3s_i-7}$ by lemma 3.3. The probability that the extended trees arising from the embedding satisfy the chosen parameters does not exceed the product of the number of different families and the bound on the probability that any given family arises. ∎

Lemma 3.6. *If $b \geq 8$, then the probability that an embedding of $\mathcal{T}$ results in a family of t extended trees in which each T_i contains m_i star nodes and has highest node v_i is at most*

$$\left(\frac{n+1}{N}\right)^{t-1}(2^{2b-7}(b+1)^{-2})^t(2^{3-b}(b+1))^m 2^{2t}$$

Proof. Let $\sum_{<l_i>}$ indicate the sum over all possible integral values of $l_1, l_2, \ldots, l_t$ and similarly for $\sum_{<s_i>}$. An upper bound on the desired probability comes from summing the bounds in lemma 3.5.

$$\sum_{<l_i>}\sum_{<s_i>}\left(\left(\frac{n+1}{N}\right)^{t-1}(2^{2b-7}(b+1)^{-2})^t(2^{3-b}(b+1))^m\prod_i(2^{4-b}(b+1))^{l_i}(2^{3-b}(b+1))^{s_i}\right)$$

$$= \left(\frac{n+1}{N}\right)^{t-1}(2^{2b-7}(b+1)^{-2})^t(2^{3-b}(b+1))^m\sum_{<l_i>}\sum_{<s_i>}\prod_i(2^{4-b}(b+1))^{l_i}(2^{3-b}(b+1))^{s_i}$$

$$= \left(\frac{n+1}{N}\right)^{t-1}(2^{2b-7}(b+1)^{-2})^t(2^{3-b}(b+1))^m\prod_i\left(\sum_{l_i}\sum_{s_i}(2^{4-b}(b+1))^{l_i}(2^{3-b}(b+1))^{s_i}\right)$$

$$= \left(\frac{n+1}{N}\right)^{t-1}(2^{2b-7}(b+1)^{-2})^t(2^{3-b}(b+1))^m\prod_i\left(\sum_{l_i}(2^{4-b}(b+1))^{l_i}\right)\left(\sum_{s_i}(2^{3-b}(b+1))^{s_i}\right)$$

$$\leq \left(\frac{n+1}{N}\right)^{t-1}(2^{2b-7}(b+1)^{-2})^t(2^{3-b}(b+1))^m\prod_i\left(\frac{16}{7}\right)\left(\frac{32}{23}\right)$$

$$\leq \left(\frac{n+1}{N}\right)^{t-1}(2^{2b-7}(b+1)^{-2})^t(2^{3-b}(b+1))^m 2^{2t}$$

∎

Lemma 3.7. *If $b \geq 8$, then the probability that an embedding of $\mathcal{T}$ results in m star nodes contained in a family of t extended trees is at most*

$$\binom{N}{t}\binom{m-1}{t-1}\left(\frac{n+1}{N}\right)^{t-1}(2^{2b-7}(b+1)^{-2})^{t}(2^{3-b}(b+1))^{m}2^{2t}$$

Proof. The first factor represents the number of ways to choose the highest node in each tree. Order does not matter when we choose the highest nodes, but we must then assign an arbitrary order to distinguish between the different extended trees. The second factor represents the number of ways to apportion the m star nodes among the extended trees so that each tree receives at least one star node. The remaining terms come from lemma 3.6. ∎

Lemma 3.8. *Fix $b \geq 8$ and let c be a sufficiently large constant. The probability that a dynamic embedding of $\mathcal{T}$ in H_n results in cn star nodes is at most*

$$\left(\frac{cn.N}{n+1}\right)\left(e^{\frac{2^{b-1}}{b+1}}\right)^{c\frac{1}{2}n}\left(2^{3-b}(b+1)\right)^{cn}$$

Proof. Examine the bound found in lemma 3.7. The first two terms are easily bounded by $(\frac{Ne}{t})^t$ and $(\frac{cne}{t})^t$, respectively. Combining like terms reduces the bound to

$$\left(\frac{ce^2 2^{2b-5}n(n+1)}{(b+1)^2t^2}\right)^{t}\left(\frac{N}{n+1}\right)\left(2^{3-b}(b+1)\right)^{cn}$$

Using elementary calculus, we find that for sufficiently large c, the first term is optimized when $t^2 = \frac{c2^{2b-5}n(n+1)}{(b+1)^2}$. Thus, over all values of t, the largest the first term can be is $(e^{\frac{2^{b-1}}{b+1}})^{c\frac{1}{2}n}$. We then add this bound over the cn possible values of t to finish the proof. ∎

Examine this last bound on the probability that there are cn star nodes. As we increase c, the third term eventually dominates the other two terms. This allows us to force the probability to be as small a power of N as we desire. This proves the theorem.

We know that, with high probability, no more than cn tree nodes are assigned to any star in the hypercube. What remains is to distribute them throughout the star so that no node has more than a constant number of tree nodes assigned to it. In the final version of this paper we will show how to remove some nodes from each star so that the partial stars remaining cover the nodes of the hypercube. Further, each node is covered by exactly one partial star and each partial star has at least $\frac{n}{2}$ nodes. The embedding is redistributed (dynamically) as follows. The first $2c$ tree nodes assigned to a hypercube node x are stored there. Any further nodes which the algorithm tries to store at x are sent to some other node in x's partial star containing fewer than $2c$ tree nodes. Such a node always exists since there are no more than cn tree nodes in the partial star at any time. This adds at most two to the dilation of the embedding.

Now suppose that a tree node which was redistributed in the star wants to grow a child in the tree. Rather than growing a path in the hypercube from the node in which it is actually

stored, the path is grown from the hypercube node which was originally supposed to contain the tree node. This may add another two to the dilation, but it is important for the analysis that the path continues to grow randomly from the end of the last random path.

The load-balancing adjustments at both ends of the path may add up to 4 to the dilation, resulting in a path length which is at most 12. Since each tree edge is mapped to a path of length no greater than 12, the dilation of the embedding is constant.

Finally, consider a tree $\mathcal{T}$ which grows and shrinks. Any form taken by $\mathcal{T}$ has been embedded according to the algorithm given above. Since there are only a polynomial number of such trees, the theorem holds in this case as well.

4 A Lower Bound for Deterministic Algorithms

The following theorem shows that determinstic algorithms perform poorly on this problem. We prove that, even if we restrict our attention to trees which only grow, any deterministic embedding algorithm which maintains load c must have not only maximum but *average* dilation $\Omega(\frac{\sqrt{\log N}}{c^2})$. A consequence of this is that any embedding algorithm which maintains constant load must necessarily have dilation $\Omega(\sqrt{\log N})$.

Theorem: Any deterministic algorithm for dynamically embedding trees in the hypercube which achieves load c must have average edge length $\Omega(\frac{\sqrt{\log N}}{c^2})$.

Proof: Let c be the load maintained by the embedding algorithm. Define the *size* of a node in the hypercube to be the number of 1's in the node's string. Partition the hypercube into 6c levels, each level corresponding to some range of node sizes and containing $\frac{N}{6c}$ nodes. Since there are at most $O(\frac{N}{\sqrt{\log N}})$ nodes of any size, each level must contain at least $\Omega(\frac{\sqrt{\log N}}{c})$ sizes. This means that any two nodes which are in non-adjacent levels are at distance $\Omega(\frac{\sqrt{\log N}}{c})$ from each other.

Grow a path of $\frac{N}{2}$ nodes, starting at the root. Then some level must contain $\frac{N}{12c}$ tree nodes: choose such a level. We will continue growing the tree from the $\frac{N}{12c}$ nodes in the chosen level. Grow paths from each of these tree nodes simultaneously, stopping each path's growth when it reaches a hypercube node which is neither in the chosen level nor in a level adjacent to it. The total number of nodes in the chosen level and adjacent levels is at most $\frac{N}{2c}$. It follows that the total length of the $\frac{N}{12c}$ paths is at most $\frac{N}{2}$ — the number of tree nodes which can be stored in just these levels. This verifies that the tree being considered has at most N nodes.

Now we can calculate the average edge length. Since each of the $\frac{N}{12c}$ paths connects a node in the chosen level to a node in some non-adjacent level, the total edge length in these paths is at least $\frac{N}{12c}\Omega(\frac{\sqrt{\log N}}{c}) = \Omega(\frac{N\sqrt{\log N}}{c^2})$. Since the entire tree contains at most N nodes, it follows that the average edge length of the embedding is $\Omega(\frac{\sqrt{\log N}}{c^2})$. ∎

5 References

[BC] S. Bhatt and J.-Y. Cai, "Take a Walk, Grow a Tree," 29th Ann. Symp. on Foundations of Computer Science, pp. 469 - 478, October 1988.

[BCLR] S.N. Bhatt, F.R.K. Chung, F.T. Leighton and A.L. Rosenberg, "Optimal simulations of tree machines," FOCS 1986.

[KZ] R.M. Karp and Y. Zhang, "A randomized parallel branch-and-bound procedure," Proceedings of the 20th STOC, 1988.